Lab Exercise 9– Creating Multiple EC2 Instances with for_each in Terraform

Objective:

Learn how to use for_each in Terraform to create multiple AWS EC2 instances withspecific settings for each instance.

Prerequisites:

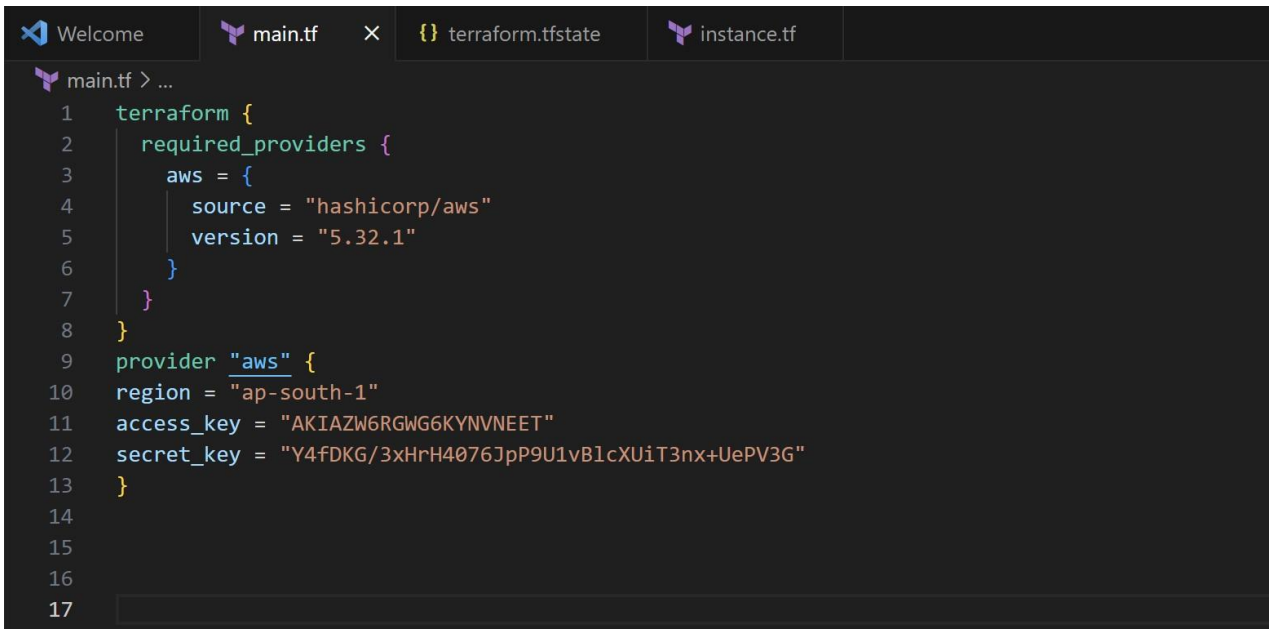
- Terraform installed on your machine.
- AWS CLI configured with the necessary credentials.

Steps:

1. Create a Terraform Directory:

- Create Terraform Configuration Files:
- Create a file named main.tf:

 # main.tf



- Replace "your-key-pair-name" and "your-subnet-id" with your actual key pair name and subnet ID.
- In this configuration, we define a variable instances as a map containing settings for each EC2 instance. The aws_instance resource is then used with for_each to create instances based on the map.

 # Instance.tf

```
instance.tf > variable "instances" > default > instance2 > instance_type
resource "aws_instance" "ec2_instances" {
  for_each = var.instances
  ami           = var.instances[each.key].ami
  instance_type = var.instances[each.key].instance_type
  tags = {
    Name = "EC2-Instance-${each.key}"
  }
}
variable "instances" {
description = "Map of EC2 instances with settings"
default = {
    "instance1" = {
      ami           = "ami-03f4878755434977f"
      instance_type = "t2.micro"
    },
    "instance2" = {
      ami           = "ami-03f4878755434977f"
      instance_type = "t2.small"
    },
    "instance3" = {
      ami           = "ami-03f4878755434977f"
      instance_type = "t2.micro"
    }
  }
}
```

2. Initialize and Apply:

- Run the following Terraform commands to initialize and apply the configuration:

terraform init

terraform apply

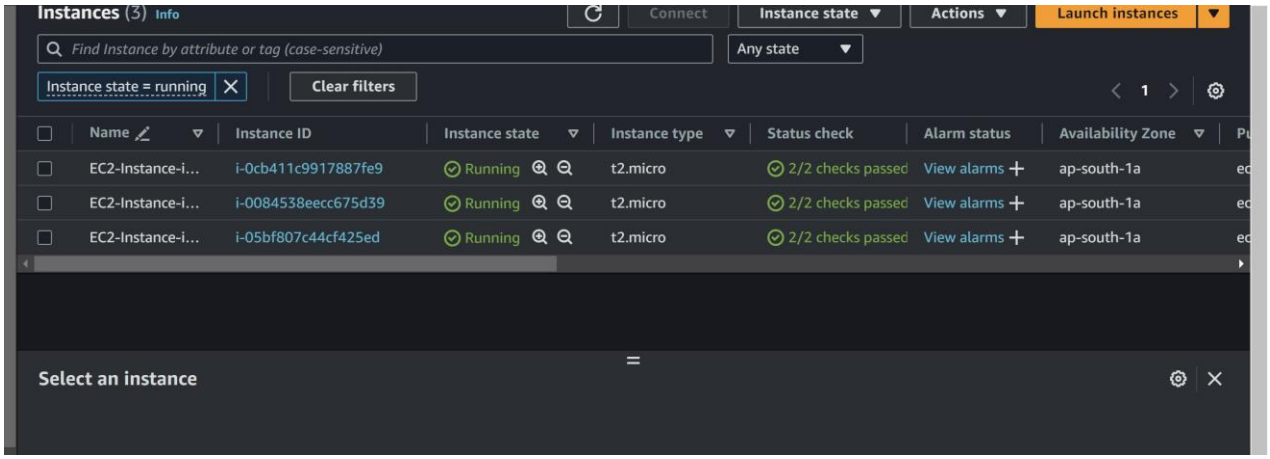
```
22
23
24

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    COMMENTS

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS E:\terraform-variables\Terraformexp9>

                                                    Column Selection    Ln 17, Col 1
```

```
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS E:\terraform-variables\Terraformexp9> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated w
  + create

Terraform will perform the following actions:

  # aws_instance.ec2_instances["instance1"] will be created
  + resource "aws_instance" "ec2_instances" {
      + ami                                  = "ami-03f4878755434977f"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    COMMENTS

  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.ec2_instances["instance3"]: Creating...
aws_instance.ec2_instances["instance2"]: Creating...
aws_instance.ec2_instances["instance2"]: Still creating... [10s elapsed]
aws_instance.ec2_instances["instance3"]: Still creating... [10s elapsed]
aws_instance.ec2_instances["instance3"]: Still creating... [20s elapsed]
aws_instance.ec2_instances["instance2"]: Still creating... [20s elapsed]
aws_instance.ec2_instances["instance3"]: Creation complete after 25s [id=i-0084538eecc
aws_instance.ec2_instances["instance2"]: Still creating... [30s elapsed]
aws_instance.ec2_instances["instance2"]: Creation complete after 35s [id=i-0cb411c9917

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
PS E:\terraform-variables\Terraformexp9>
```

- Terraform will prompt you to confirm the creation of EC2 instances. Type yes andpress Enter.

3. Verify Instances in AWS Console:

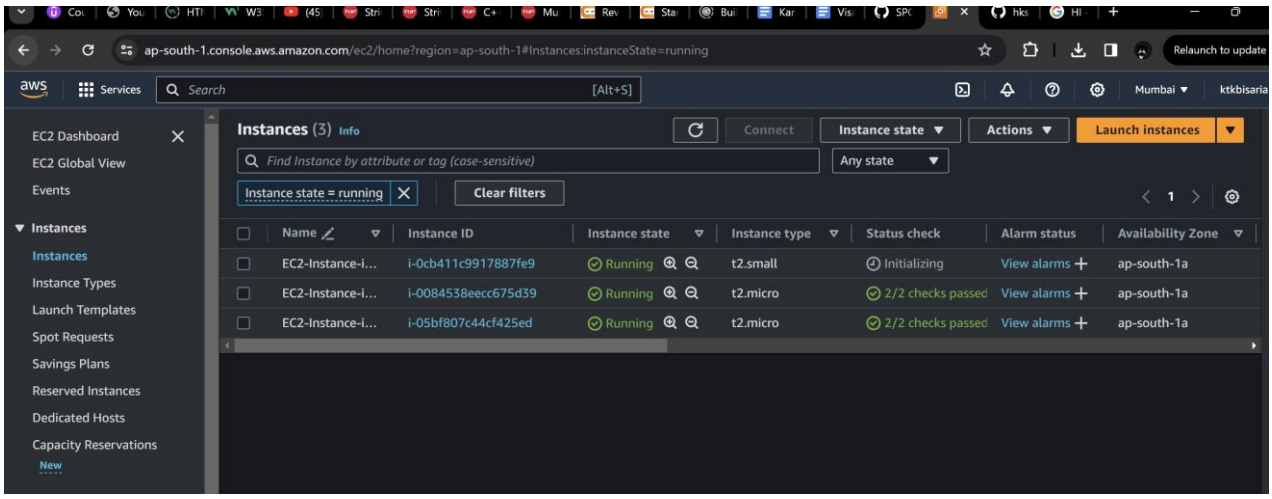
- Log in to the AWS Management Console and navigate to the EC2 service.
- Verify that the specified EC2 instances with the specified names and settings havebeen created.
- 

4. Update Instance Configuration:

- If you want to modify the EC2 instance configuration, update the main.tf file withthe desired changes.
- Rerun the terraform apply command to apply the changes:

terraform apply

```
  Enter a value: yes

aws_instance.ec2_instances["instance2"]: Modifying... [id=i-0cb411c9917887f
aws_instance.ec2_instances["instance2"]: Still modifying... [id=i-0cb411c99
aws_instance.ec2_instances["instance2"]: Still modifying... [id=i-0cb411c99
aws_instance.ec2_instances["instance2"]: Still modifying... [id=i-0cb411c99
aws_instance.ec2_instances["instance2"]: Still modifying... [id=i-0cb411c99
aws_instance.ec2_instances["instance2"]: Still modifying... [id=i-0cb411c99
aws_instance.ec2_instances["instance2"]: Still modifying... [id=i-0cb411c99
aws_instance.ec2_instances["instance2"]: Modifications complete after 1m6s

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
PS E:\terraform-variables\Terraformexp9>
```



5. Clean Up:

- After testing, you can clean up the EC2 instances:

terraform destroy

```
aws_instance.ec2_instances["instance2"]: Destroying... [id=i-0cb411c9917887fe9]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-0cb411c9917887fe9, 10s elapsed]
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-0084538eecc675d39, 10s elapsed]
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-05bf807c44cf425ed, 10s elapsed]
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-05bf807c44cf425ed, 20s elapsed]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-0cb411c9917887fe9, 20s elapsed]
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-0084538eecc675d39, 20s elapsed]
aws_instance.ec2_instances["instance1"]: Still destroying... [id=i-05bf807c44cf425ed, 30s elapsed]
aws_instance.ec2_instances["instance3"]: Still destroying... [id=i-0084538eecc675d39, 30s elapsed]
aws_instance.ec2_instances["instance2"]: Still destroying... [id=i-0cb411c9917887fe9, 30s elapsed]
aws_instance.ec2_instances["instance3"]: Destruction complete after 31s
aws_instance.ec2_instances["instance2"]: Destruction complete after 31s
aws_instance.ec2_instances["instance1"]: Destruction complete after 31s

Destroy complete! Resources: 3 destroyed.
PS E:\terraform-variables\Terraformexp9>
```

- Confirm the destruction by typing yes.

6. Conclusion:

This lab exercise demonstrates how to use the for_each construct in Terraform to create multiple AWS EC2 instances with specific settings for each instance. The use ofa map allows you to define and manage settings for each instance individually.

Experiment with different instance types, AMIs, and settings in the main.tf file toobserve how Terraform provisions resources based on your configuration.